Reliable Service Allocation in Clouds with Memory and Capacity Constraints

Olivier Beaumont[1], Lionel Eyraud-Dubois[1],
Pierre Pesneau[2], and Paul Renaud-Goud[1]

[1] Inria & University of Bordeaux – Bordeaux, France
{olivier.beaumont, lionel.eyraud-dubois,
paul.renaud-goud}@inria.fr
[2] University of Bordeaux & Inria – Bordeaux, France
{pierre.pesneau}@inria.fr

Abstract. We consider allocation problems that arise in the context of service allocation in Clouds. More specifically, on the one part we assume that each Physical Machine (denoted as PM) is offering resources (memory, CPU, disk, network). On the other part, we assume that each application in the IaaS Cloud comes as a set of services running as Virtual Machines (VMs) on top of the set of PMs. In turn, each service requires a given quantity of each resource on each machine where it runs (memory footprint, CPU, disk, network). Moreover, there exists a Service Level Agreement (SLA) between the Cloud provider and the client that can be expressed as follows: the client requires a minimal number of service instances which must be alive at the end of the day, with a given reliability (that can be converted into penalties paid by the provider). In this context, the goal for the Cloud provider is to find an allocation of VMs onto PMs so as to satisfy, at minimal cost, both capacity and reliability constraints for each service. In this paper, we propose a simple model for reliability constraints and we prove that it is possible to derive efficient heuristics.

Keywords: Cloud, reliability, resilience, linear programming, VM allocation, multidimensional bin packing

1 Introduction

1.1 Reliability and Energy Savings in Cloud Computing

In this paper, we consider reliability issues in the context of Cloud Computing platforms, when allocating a set of independent services running as Virtual Machines (VMs) onto Physical Machines (PMs) in a Cloud computing platform. Cloud Computing [19,1] is a well-suited paradigm for service providing over the Internet. Using virtualization, it is possible to run several Virtual Machines on top of a given Physical Machine. Since each VM hosts its complete software

stack (Operating System, Middleware, Application), it is moreover possible to migrate VMs from a PM to another in order to dynamically balance the load.

Even in the static case, mapping VMs with heterogeneous computing demands onto PMs with capacities is amenable to a multi-dimensional bin-packing problem. Indeed, in this context, each physical machine comes with its computing capacity (*i.e.* the number of flops it can process during one time-unit), its disk capacity (*i.e.* the number of bytes it can read/write during one time-unit), its network capacity (*i.e.* the number of bytes it can send/receive during one time-unit), its memory capacity (given that each VM comes with its complete software stack) and its failure rate (*i.e.* the probability that the machine will fail during the next time period).

On the other hand, each service comes with its requirement along the same dimensions (memory, CPU, disk and network footprints) and a reliability demand that has been negociated through the SLA [9]. In order to deal with capacity constraints in resource allocation problems, several sophisticated techniques have been developed in order to optimally allocate VMs onto PMs, either to achieve good load balancing [18,3] or to minimize energy consumption [5,4]. Most of the works in this domain have therefore concentrated on designing offline [12] and online [13] Packing variants.

In this paper, we propose to use a new set of variables in order to deal with this multi-dimensional bin-packing problem. More specifically, we use the specific structure of the problem, and in particular the fact that the number of VMs a given PM can host is very small in practice, since each VM comes with its complete software stack. Therefore, our approach consists in enumerating all possible configurations (a configuration being defined by the set of services running on a machine) and using them as variables in the optimization problems. Indeed, if the number of different services a PM can simultaneously host is bounded, then the number of possible configurations has polynomial size.

Reliability constraints have received much less attention in the context of Cloud computing, as underlined by Cirne *et al.* [9]. Nevertheless, related questions have been addressed in the context of more distributed and less reliable systems such as Peer-to-Peer networks. In such systems, efficient data sharing is complicated by erratic node failure, unreliable network connectivity and limited bandwidth. Thus, data replication can be used to improve both availability and response time and the question is to determine where to replicate data in order to meet performance and availability requirements in large-scale systems [15,17,16]. Reliability issues have also been addressed by High Performance Computing community. Indeed, recently, a lot of effort has been done to build systems capable of reaching the Exaflop performance [10] and such exascale systems are expected to gather billions of processing units, thus increasing

the importance of fault tolerance issues [8]. Solutions for fault tolerance in Exascale systems are based on replication strategies [11] and rollback recovery relying on checkpointing protocols [6].

This work is a follow-up of [2], where the question of how to evaluate the reliability of an allocation has been addressed and a set of deterministic and randomized heuristics have been proposed in the context of a single type of constraint related to CPU usage. Moreover, in [2], in order to avoid difficulties related to the bin packing problem, we assumed that the overall CPU demand of a service could be split arbitrarily on an arbitrary number of machines. Under this assumption, the packing problem was equivalent to a mono-dimensional splittable item bin packing problem, that can be solved in polynomial time without cardinality constraints.

In this paper, we therefore consider a more general model and we rely on the additional assumption on the number of VMs running on a PM in order to obtain tractable formulations of the optimization problem, that consist, from the provider point of view, to obtain an allocation of services to physical machines so as to satisfy both capacity and reliability constraints.

1.2 Model and Notations

In this section, we introduce the notations that will be used throughout the paper. Our target cloud platform is made of m physical machines $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$. We consider the behavior of this platform over the course of one given time period (for example over one day), and we assume that during that time period, it is not practical to reallocate instances of services to physical machines. The goal of our allocation procedure is thus to provision extra instances (replicas) for the services, which will actually be used if some machines fail during the time period. The success of the allocation is determined at the end of the time period: a service is successful if enough instances are still running at the end of the time period.

As already noted, machine $\mathcal{M}_j$ is characterized by its capacity along several dimensions, *i.e.* memory, CPU, disk, network,... Since memory plays a specific role in our model, we consider it separately, and we note K the number of other resources. For the sake of simplicity, we assume that machines are homogeneous (extending this work to a constant number of machine classes is possible) and we will denote by MEMCAPA the memory capacity of the machines, and by $\text{CAPA}_k,\ 1 \leq k \leq K$ the capacities of the machines along other dimensions. Moreover, we consider a given failure probability FAIL, that represents the probability of failure of any given machine during the time period.

On this Cloud platform, we have to run n services $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$. DEM_i identical and independent instances of service $\mathcal{S}_i$ are required, and the instances

of the different services run as Virtual Machines. In turn, service $\mathcal{S}_i$ is characterized by its capacity along several dimensions, *i.e.* memory, CPU, disk, network,... We will denote by MEMPRINT_i the memory footprint of service $\mathcal{S}_i$ and by $\text{PRINT}_{k,i},\ 1 \leq k \leq K$ the footprints of service $\mathcal{S}_i$ along the other dimensions. Several instances of the same service can be run concurrently and independently on the same physical machine and we will denote by $\mathcal{A}_{i,j}$ the number of instances of $\mathcal{S}_i$ running on $\mathcal{M}_j$.

Therefore, $\sum_i \mathcal{A}_{i,j}\text{MEMPRINT}_i$ represents the overall memory footprint of the services running on $\mathcal{M}_j$ and therefore, it has to be smaller than $\text{MEMCAPA},\ \forall j$. The same applies for the other resources and in general $\forall j, k\ , \sum_i \mathcal{A}_{i,j}\text{PRINT}_{k,i} \leq \text{CAPA}_k$.

On the other hand, $\sum_j \mathcal{A}_{i,j}$ represents the overall number of running instances of $\mathcal{S}_i$. In general, $\sum_j \mathcal{A}_{i,j}$ is larger than DEM_i since we use replication (*i.e.* over-provisioning) of services in order to enforce the reliability requirements. We will denote by ALIVE the set of machines which are still running at the end of the time period. In our model, at the end of the time period, the machines are either up or completely down, so that the number of instances of Service $\mathcal{S}_i$ running on $\mathcal{M}_j$ is $\mathcal{A}_{i,j}$ if $\mathcal{M}_j \in \text{ALIVE}$ and 0 otherwise. Therefore, $\text{ALIVEINST}_i = \sum_{j,\mathcal{M}_j \in \text{ALIVE}} \mathcal{A}_{i,j}$ denotes the overall number of running instances of $\mathcal{S}_i$ at the end of the time period and $\mathcal{S}_i$ is running properly at the end of the time period if and only if $\sum_{j,\mathcal{M}_j \in \text{ALIVE}} \mathcal{A}_{i,j} \geq \text{DEM}_i$.

Of course, our goal is not to guarantee that all services should run properly at the end of the time period with probability 1. Indeed, such a reliability cannot be achieved in practice since the probability that all machines fail is clearly larger than 0 in our model. In general, as noted in a recent paper of the NY Times [14], Data Centers usually over-provision resources (at the price of high energy consumption) in order to (quasi-)avoid failures. In our model, we assume a more sustainable model, where the SLA defines the reliability requirement REL_i for Service $\mathcal{S}_i$ (together with the penalty paid by the Cloud Provider if $\mathcal{S}_i$ does not run with at least DEM_i instances at the end of the period). Therefore, the Cloud provider faces the following optimization problem:

ValidAllocation$(m, n, \text{DEM}, \text{REL})$: Find an allocation $\mathcal{A}$ of the instances of services $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ to machines $\mathcal{M}_1, \mathcal{M}_2, \ldots \mathcal{M}_m$ such that

- **Packing:** $\begin{cases} \forall j, \sum_i \mathcal{A}_{i,j}\text{MEMPRINT}_i \leq \text{MEMCAPA} & \text{and} \\ \forall j, k\ , \sum_i \mathcal{A}_{i,j}\text{PRINT}_{k,i} \leq \text{CAPA}_k. \end{cases}$
- **Reliability:** $\forall i,\ \mathcal{P}(\text{ALIVEINST}_i \geq \text{DEM}_i) \geq \text{REL}_i$, *i.e.* the probability that a least DEM_i instances of $\mathcal{S}_i$ are running on alive machines after the time period is larger than the reliability requirement REL_i.

1.3 Outline of the Paper

As we have noticed it, **ValidAllocation** is in general a difficult problem. First, it has been proved in [2] that given an allocation $\mathcal{A}$, determining if Reliability constraints are enforced is in general #P-Complete. Moreover, in its general form, the Packing problem is a multi-dimensional bin packing problem that is NP-Complete and hard to approximate. Therefore, in Section 2, we propose a different model that takes into account the specific characteristics of the solution in order to obtain a formulation of the optimization problem that can be given to an optimization solver and produce a solution in a small amount of time. The results obtained on a characteristic set of instances are given in Section 3 and the conclusions and perspectives in Section 4

2 Reformulation of the Optimization problem

In this section, we propose a different formulation of the optimization problem **ValidAllocation**. In Section 2.1, we formulate the optimization problem for a single service as a non-linear optimization problem that can be solved efficiently under the assumption that we restrict the search to almost homogeneous allocations. Then, in Section 2.2, we rely on the fact that in practice, the number of services that can be run simultaneously on a given PM is usually very small, so that it is possible to enumerate all possible valid allocation schemes for a machine. Moreover, we will assume that the demand DEM_i of the different services is large, so that we can accurately approximate a binomial distribution by a normal distribution when estimating the number of alive instances of a service at the end of the day.

2.1 Case of a Single Service

ValidAllocation is in general difficult to solve for two different reasons, *i.e.* (i) estimating the reliability of a single heterogeneous allocation is in general #P-Complete and (ii) with several services, it is equivalent to a multi-dimensional bin packing problem. The question related to bin packing will be addressed in Section 2.2 and we concentrate in this section on the case of a single service.

Without reliability constraints, the minimal number of resources in order to run Service $\mathcal{S}_i$ whose demand is DEM_i, whose footprint along dimension k is $\text{PRINT}_{k,i}$ and whose memory footprint is MEMPRINT_i can be found with:

$$\text{Minimize } \sum_j \mathcal{A}_{i,j} \text{ such that } \begin{cases} \forall j, & \mathcal{A}_{i,j}\text{MEMPRINT}_i \leq \text{MEMCAPA} \\ \forall j,k, & \mathcal{A}_{i,j}\text{PRINT}_{k,i} \leq \text{CAPA}_k \\ \forall j\ , & \mathcal{A}_{i,j} \geq 0 \end{cases}$$

where equivalently, if $A_{\max,i} = \min(\min_k(\frac{\text{CAPA}_k}{\text{PRINT}_{k,i}}), \frac{\text{MEMCAPA}}{\text{MEMPRINT}_i})$, all capacity constraints can be written as $\forall j, 0 \leq \mathcal{A}_{i,j} \leq A_{\max,i}$. Recall that as a number of instances of a service, all $\mathcal{A}_{i,j}$ is an integer.

In order to incorporate the set of constraints related to reliability in the above IP (Integer Program), we rely on the approximation of the binomial distribution $\mathcal{B}(N,p)$ by the normal distribution $\mathcal{N}(Np, Np(1-p))$ that is valid as soon as N is large [7] (typically, the approximation is considered as valid as soon as both Np and $N(1-p)$ are larger than 5, what is easily satisfied in our setting).

For $l < A_{\max,i}$, let us denote by n_l^i the number of machines where exactly l instances of service $\mathcal{S}_i$ are allocated. By the assumption of our model, the number of alive machines among those n_l^i machines follows a binomial distribution: $\mathcal{B}(n_l^i, p)$ where $p = 1 - \text{FAIL}$. By approximating this with a normal distribution, we get that the number of alive services on those machines is distributed as $l \times \mathcal{N}(pn_l^i, p(1-p)n_l^i) = \mathcal{N}(pln_l^i, p(1-p)l^2 n_l^i)$. Summing over l, we obtain the distribution of the number of alive instances: $\text{ALIVEINST}_i \sim \mathcal{N}(p\sum_l ln_l^i, p(1-p)\sum_l l^2 n_l^i)$. Then, let us denote by z_{REL_i} the value (that depends on REL_i only, and can be obtained in a normal distribution table) such that $\mathcal{P}(\text{ALIVEINST}_i \geq \text{DEM}_i) \geq \text{REL}_i$ as soon as $p\sum_l ln_l^i - z_{\text{REL}_i}\sqrt{p(1-p)\sum_l l^2 n_l^i} \geq \text{DEM}_i$. Therefore, the following optimization problem provides a valid allocation that minimizes the number of replicas used for Service A:

$$\text{Minimize } \sum_j \mathcal{A}_{i,j} \text{ s. t. } \begin{cases} p\sum_j \mathcal{A}_{i,j} - z_{\text{REL}_i}\sqrt{p(1-p)}\sqrt{\sum_j \mathcal{A}_{i,j}^2} \geq \text{DEM}_i \\ \forall j,\ 0 \leq \mathcal{A}_{i,j} \leq A_{\max,i} \\ \forall j\ ,\ \mathcal{A}_{i,j} \geq 0 \end{cases}$$

The constraint $p\sum_j \mathcal{A}_{i,j} - z_{\text{REL}_i}\sqrt{p(1-p)}\sqrt{\sum_j \mathcal{A}_{i,j}^2} \geq \text{DEM}_i$ states both that enough resources should be allocated to $\mathcal{S}_i$ and that the dispersion of these resources over PMs should be small. This corresponds well to what we observed in [2] (in the case without memory constraints), where homogeneous allocations were considered. Therefore, we will restrict the search to quasi-homogeneous allocations, where the number of instances $\mathcal{A}_{i,j}$ on machine j can be either $\mathcal{A}_i$ or $\mathcal{A}_i + 1$ for some value of $\mathcal{A}_i < A_{\max,i}$.

In order to determine the best (in terms on number of required resources) quasi-homogeneous allocation, the first step is to determine $\mathcal{A}_i$. $\mathcal{A}_i$ should be such that allocating $\mathcal{A}_i$ instances to all m machines should not be enough to enforce reliability constraint, but that allocating $\mathcal{A}_i+1$ resources to all m machines should be enough, *i.e.* $\mathcal{A}_i = \left\lfloor \frac{\text{DEM}_i}{pm - z_{\text{REL}_i}\sqrt{p(1-p)}\sqrt{m}} \right\rfloor$.

Then, it is easy to see that the number m_i of machines that receive $\mathcal{A}_i + 1$ instances (the other $m - m_i$ machines being allocated $\mathcal{A}_i$ instances) needs to be

at least $\lceil x \rceil$, where x satisfies the equation

$$pm\mathcal{A}_i + px - z_{\text{REL}_i}\sqrt{p(1-p)}\sqrt{px(\mathcal{A}_i+1)^2 + p(m-x)(\mathcal{A}_i)^2} = \text{DEM}_i,$$

which is equivalent to the following second order equation:

$$(pm\mathcal{A}_i - \text{DEM}_i - px)^2 - z^2_{\text{REL}_i}p(1-p)\left(px(2\mathcal{A}_i+1) + pm{\mathcal{A}_i}^2\right) = 0.$$

2.2 Enumerating the Set of Valid Allocations

As already noted in the introduction, **ValidAllocation** is in general difficult since even in absence of failures, it is equivalent to a general multi-dimensional bin packing problem, that is known to be hard to solve and to approximate. On the other hand, the dimension that corresponds to the memory constraints has several characteristics that make it suitable for optimization. Indeed, since each VM comes with its full software stack (say, a few Gigabytes), the number of VMs that can be handled simultaneously on a given PM is relatively low. Therefore, one can assume that no PM will be able to run more than $n_{\max}$ services and we will treat $n_{\max}$ as a constant (the reader should have in mind that a typical value for $n_{\max}$ is 3 or 4).

Let us denote as a configuration $\mathcal{C}_c$ a set of $n_{\max}$ services $\mathcal{S}_{c_1}, \mathcal{S}_{c_2}, \ldots, \mathcal{S}_{c_{n_{\max}}}$. We assume that each machine runs exactly $n_{\max}$ services without loss of generality, by adding a fictitious service whose footprint in any direction is 0. To be valid, configuration $\mathcal{C}_c$ must satisfy $\sum_{i=1}^{n_{\max}} \text{MEMPRINT}_{c_i} \leq \text{MEMCAPA}$ and $\forall j, k$, $\sum_{i=1}^{n_{\max}} \text{PRINT}_{k,c_i} \leq \text{CAPA}_k$. Let us now denote by $\mathcal{S}$ the set of possible valid configurations. There are at most a polynomial size number $n^{n_{\max}}$ of such configurations and the validity of each configuration can be verified in constant time. Using $\mathcal{S}$, it is possible to write a linear program (LP) that finds a valid allocation that makes use of a minimal number of machines. In the following LP, λ_c denotes the number of machines that follow configuration $\mathcal{C}_c$.

$$\text{Minimize } \sum_c \lambda_c \text{ such that } \begin{cases} \forall i, & \sum_{c,\ \#\{\mathcal{S}_i \in \mathcal{C}_c\} = \mathcal{A}_i + 1} \lambda_c = m_i \\ \forall i, & \sum_{c,\ \#\{\mathcal{S}_i \in \mathcal{C}_c\} = \mathcal{A}_i} \lambda_c = m - m_i \\ \forall c, & \lambda_c \geq 0 \end{cases}$$

3 Numerical Results

In this section, we describe an experimental analysis of the homogeneous heuristic, obtained from randomly generated scenarios.

3.1 Experimental setting

We consider an experimental setting with two dimensions of resource which represent memory and CPU usage. The memory footprints of all services are assumed to be approximately the same, and we consider that the available memory at each machine is enough to run 4 service instances. On the other hand, the CPU usage of each service is generated uniformly at random between 1% and MAXCPU $\%$ of the CPU capacity of the machines. For the parameter MAXCPU, we use three different values: 100%, 50%, and 25%, which implies that the CPU capacity of each machine is enough to run (on average) 2, 4, and 8 service instances. This allows us to consider a wide range of scenarios: when MAXCPU is 100%, the most constrained resource is CPU, whereas when MAXCPU is 25%, the most constrained resource is memory. For the intermediate case where MAXCPU is 50%, both resources have the same importance. The failure probability of each machine is set to 0.01, and the reliability requirements of each service is randomly generated in the following way: a random number is picked uniformly between 4 and 16, and the reliability requirement is set to $1 - 10^{-x/2}$ (hence reliability requirements lie between 0.99 and 0.99999999).

We generate small scenarios with $n = 20$ services, and large scenarios with $n = 50$ services. For each value of the parameters n and MAXCPU, we generate 100 scenarios. For each scenario, we compute, as a lower bound, the number of bins necessary to provide enough capacity to hold all services, without any packing constraint. We also run the homogeneous heuristic by solving the integer version of the linear program with CPLEX with a 10 second time limit. In most cases, CPLEX was able to find an optimal integral solution, and for the largest scenarios the best solution found in 10 seconds was used.

3.2 Results

Experimental results are shown on Figure 1, where we give boxplots for the ratio of the number of bins used by the homogeneous heuristic to the lower bound.

We can see that our heuristic gives very efficient solutions, especially in the cases where memory is the strongest constraint (MAXCPU $= 25\%$). When MAXCPU $= 100\%$, it is more difficult to use all of the CPU capacity of the machines, and the performance of the homogeneous heuristic is not as close to the lower bound. But even in that case, it uses on average 10% more bins than the lower bound, and most of the time only 20% more. This shows that it is possible to provide efficient packing solutions that exhibit good reliability guarantees.

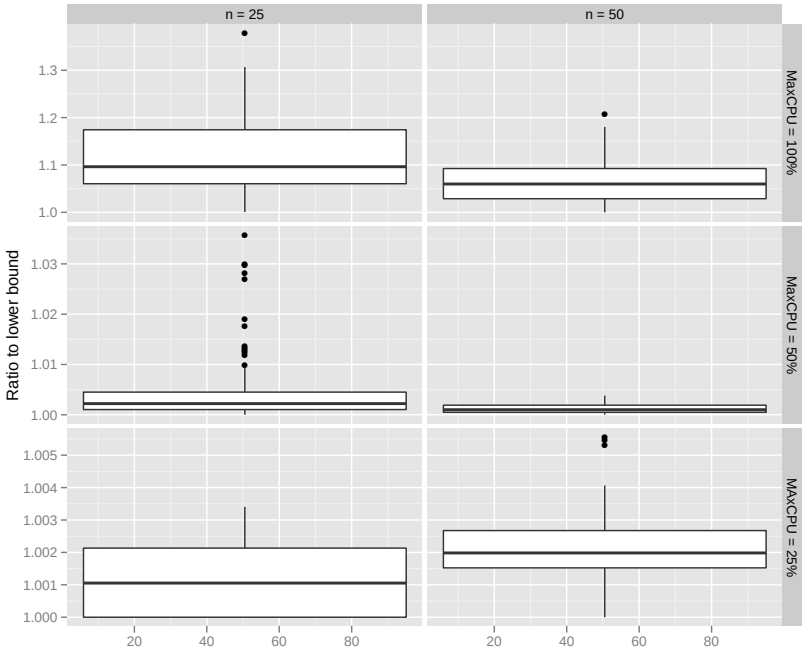

Fig. 1. Experimental results with $n = 20$ and $n = 50$ services

4 Conclusion

We have considered the difficulty introduced by reliability constraints when considering service allocation in Clouds in presence of failures. In this context, replication (or equivalently over-subscription) is used in order to enforce reliability demand. We have proposed a new formulation of the optimization based on a relaxation of an approximation of probability distributions that enables the design of a sophisticated heuristic, that turns out to be very efficient in a large number of scenarios.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: Above the clouds: A berkeley view of cloud computing. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28 (2009)
2. Beaumont, O., Eyraud-Dubois, L., Larchevêque, H.: Reliable Service Allocation in Clouds. In: IPDPS 2013 - 27th IEEE International Parallel & Distributed Processing Symposium. Boston, États-Unis (2013), `http://hal.inria.fr/hal-00743524`

3. Beaumont, O., Eyraud-Dubois, L., Rejeb, H., Thraves, C.: Heterogeneous Resource Allocation under Degree Constraints. IEEE Transactions on Parallel and Distributed Systems (2012)
4. Beloglazov, A., Buyya, R.: Energy efficient allocation of virtual machines in cloud data centers. In: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. pp. 577–578. IEEE (2010)
5. Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M., Pentikousis, K.: Energy-efficient cloud computing. The Computer Journal 53(7), 1045 (2010)
6. Bougeret, M., Casanova, H., Rabie, M., Robert, Y., Vivien, F.: Checkpointing strategies for parallel jobs. In: High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for. pp. 1–11. IEEE (2011)
7. Box, G.E., Hunter, W.G., Hunter, J.S.: Statistics for experimenters: an introduction to design, data analysis, and model building (1978)
8. Cappello, F.: Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities. International Journal of High Performance Computing Applications 23(3), 212–226 (2009)
9. Cirne, W., Frachtenberg, E.: Web-scale job scheduling. In: Job Scheduling Strategies for Parallel Processing. pp. 1–15. Springer (2013)
10. Dongarra, J., Beckman, P., Aerts, P., Cappello, F., Lippert, T., Matsuoka, S., Messina, P., Moore, T., Stevens, R., Trefethen, A., et al.: The international exascale software project: a call to cooperative action by the global high-performance community. International Journal of High Performance Computing Applications 23(4), 309–322 (2009)
11. Ferreira, K., Stearley, J., Laros III, J., Oldfield, R., Pedretti, K., Brightwell, R., Riesen, R., Bridges, P., Arnold, D.: Evaluating the viability of process replication reliability for exascale systems. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. p. 44. ACM (2011)
12. Garey, M.R., Johnson, D.S.: Computers and Intractability, a Guide to the Theory of NP-Completeness. W. H. Freeman and Company (1979)
13. Hochbaum, D.: Approximation Algorithms for NP-hard Problems. PWS Publishing Company (1997)
14. Data centers waste vast amounts of energy belying industry image. http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html
15. Ranganathan, K., Iamnitchi, A., Foster, I.: Improving data availability through dynamic model-driven replication in large peer-to-peer communities. In: Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on. p. 376 (may 2002)
16. Santos-Neto, E., Cirne, W., Brasileiro, F., Lima, A.: Exploiting replication and data reuse to efficiently schedule data-intensive applications on grids. In: Feitelson, D., Rudolph, L., Schwiegelshohn, U. (eds.) Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol. 3277, pp. 54–103. Springer Berlin / Heidelberg (2005)
17. da Silva, D., Cirne, W., Brasileiro, F.: Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.) Euro-Par 2003 Parallel Processing, Lecture Notes in Computer Science, vol. 2790, pp. 169–180. Springer Berlin / Heidelberg (2003)
18. Van, H., Tran, F., Menaud, J.: SLA-aware virtual resource management for cloud infrastructures. In: IEEE Ninth International Conference on Computer and Information Technology. pp. 357–362. IEEE (2009)
19. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications 1(1), 7–18 (2010)